Universidad de San Carlos de Guatemala

Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas Manejo e Implementación de Archivos
Segundo Semestre 2021
Catedráticos: Ing. Álvaro Díaz, Ing. Oscar Paz, Ing. William Escobar
Tutores académicos: Bruno Coronado, Renato Flores, Elizabeth Zavala, Alan Morataya

Proyecto 1

Introducción:

El curso de Manejo e Implementación de Archivos busca que el estudiante aprenda los conceptos sobre la administración de archivos, tanto en hardware como software, sistemas de archivos, particiones, entre otros conceptos, para después introducirse en las bases de datos. El proyecto busca que el estudiante aplique estos conceptos y pueda aprenderlos implementando su funcionalidad.

Objetivos:

- Aprender a administrar archivos y escribir estructuras en C/C++
- Comprender el sistema de archivos EXT3 y EXT2
- Aplicar el formateo rápido y completo en una partición
- Crear una aplicación de comandos

- Aplicar la teoría de ajustes
- Aplicar la teoría de particiones
- Utilizar GraphViz para mostrar reportes
- Restringir y administrar el acceso a los archivos y carpetas en ext3/ext2 por medio de usuarios
- Administrar los usuarios y permisos por medio de grupo

Aplicación de Comandos:

La aplicación será totalmente en consola, a excepción de los reportes en Graphviz. Esta no tendrá menús, sino que se utilizarán comandos. No distinguirá entre mayúsculas y minúsculas. La aplicación será capaz de leer desde standard input(stdin) y escribir a standard output (stdout) de ser necesario. Hay parámetros obligatorios y opcionales. Solo se puede colocar un comando por línea.

Si se utiliza un parámetro que no está especificado en este documento, debe mostrar un mensaje de error. Se utilizarán espacios en blanco para separar cada parámetro. Si se necesita que algún valor lleve espacios en blanco se encerrará entre comillas " ". **Los parámetros pueden venir en cualquier orden.**

ADMINISTRACIÓN DE DISCOS:

Estos comandos permitirán crear archivos que simularán discos duros en los que se podrá formatear más adelante con el sistema de archivos ext2 o ext3. Estos comandos estarán disponibles desde que se inicia el programa. Estos comandos son:

1. MKDISK

Este comando creará un archivo binario que simulará un

disco duro, estos archivos binarios tendrán la extensión **disk** y su contenido al inicio será 0. Deberá ocupar físicamente el tamaño indicado por los parámetros, (no importa que el sistema operativo no muestre el tamaño exacto). Recibirá el nombre del archivo que simulará el disco duro y tendrá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-size	Obligatorio	Este parámetro recibirá un número que indicará el tamaño del disco a crear. Debe ser positivo y mayor que cero, si no se mostrará un error.
-f	Opcional	Indicará el ajuste que utilizará el disco para crear las particiones dentro del disco Podrá tener los siguientes valores: BF: Indicará el mejor ajuste (Best Fit) FF: Utilizará el primer ajuste (First Fit) WF: Utilizará el peor ajuste (Worst Fit) Ya que es opcional, se tomará el primer ajuste si no está especificado en el comando. Si se utiliza otro valor que no sea alguno de los anteriores mostrará un mensaje de error.

-u	Opcional	Este parámetro recibirá una letra que indicará las unidades que utilizará el parámetro size. Podrá tener los siguientes valores: **k** que indicará que se utilizarán Kilobytes (1024 bytes) **m** en el que se utilizarán Megabytes (1024 * 1024 bytes) Este parámetro es opcional, si no se encuentra se creará un disco con tamaño en Megabytes. Si se utiliza otro valor debe mostrarse un mensaje de error.
-path	Obligatorio	Este parámetro será la ruta en el que se creará el archivo que representará el disco duro. Si las carpetas de la ruta no existen deberán crearse.

Ejemplos:

```
#Crea un disco de 3000 Kb en la carpeta  home
Mkdisk -Size=3000 –u=K -path=/home/user/Disco1.dk

#No es necesario utilizar comillas para la ruta en este caso ya que la ruta
 no tiene ningún espacio en blanco
Mkdisk –path=/home/user/Disco2.dk –U=K –size=3000

#Se ponen comillas por la carpeta “mis discos” ya que tiene espacios en
 blanco, se crea si no  existe.
mkdisk –size=5 –u=M –path="/home/mis discos/Disco3.dk"
```

```
#Creará un disco de 10 Mb ya que no hay parámetro unit
Mkdisk –size=10 –path="/home/mis discos/Disco4.dk"
```

2. RMDISK

Este parámetro elimina un archivo que representa a un disco duro mostrando un mensaje de confirmación para eliminar. Tendrá los siguientes parámetros:

Parámetro	Categoría	Descripción
-path	Obligatorio	Este parámetro será la ruta en el que se eliminará el archivo que representará el disco duro. Si el archivo no existe, debe mostrar un mensaje de error.

Ejemplo:

```
#Elimina Disco4.dk
rmDisk -path="/home/mis discos/Disco4.dk"
```

3. FDISK

Este comando administra las particiones en el archivo que representa al disco duro. Deberá mostrar un error si no se pudo realizar la operación solicitada sobre la partición, especificando por qué razón no pudo crearse (Por espacio, por restricciones de particiones, etc.).

No se considerará el caso de que se pongan parámetros incompatibles, por ejemplo, en un mismo comando fdisk llamar a delete y add. La estructura de cada disco se explicará más adelante. Tendrá los siguientes parámetros:

Parámetro	Categoría	Descripción
-size	Obligatorio al crear	Este parámetro recibirá un número que indicará el tamaño de la partición a crear. Debe ser positivo y mayor a cero, de lo contrario se mostrará un mensaje de error.
-u	Opcional	Este parámetro recibirá una letra que indicará las unidades que utilizará el parámetro size. Podrá tener los siguientes valores: **B:** indicará que se utilizarán bytes. **K:** indicará que se utilizarán Kilobytes(1024 bytes) **M:**indicará que se utilizarán Megabytes(1024 * 1024 bytes). Este parámetro es opcional, si no se encuentra se creará una partición en Kilobytes. Si se utiliza un valor diferente mostrará un mensaje de error.
-path	Obligatorio	Este parámetro será la ruta en la que se encuentra el disco en el que se creará la partición. Este archivo ya debe existir, si no se mostrará un error.
-type	Opcional	Indicará que tipo de partición se creará. Ya que es opcional, se tomará como primaria en caso de que no se indique. Podrá tener los siguientes valores: **P:** en este caso se creará una partición primaria. **E:** en este caso se creará una partición extendida. **L:** Con este valor se creará una partición lógica. Las particiones lógicas sólo pueden

		estar dentro de la extendida sin sobrepasar su tamaño. Deberá tener en cuenta las restricciones de teoría de particiones: La suma de primarias y extendidas debe ser como máximo 4. Solo puede haber una partición extendida por disco. No se puede crear una partición lógica si no hay una extendida. Si se utiliza otro valor diferente a los anteriores deberá mostrar un mensaje de error.
-f	Opcional	Indicará el ajuste que utilizará la partición para asignar espacio. Podrá tener los siguientes valores: **BF:** Indicará el mejor ajuste (Best Fit) **FF:** Utilizará el primer ajuste (First Fit) **WF:** Utilizará el peor ajuste (Worst Fit) Ya que es opcional, se tomará el peor ajuste si no está especificado en el comando. Si se utiliza otro valor que no sea alguno de los anteriores mostrará un mensaje de error.
-delete	Opcional	Este parámetro indica que se eliminará una partición. Este parámetro se utiliza junto con **-name** y **-path**. Se deberá mostrar un mensaje que permita confirmar la eliminación de dicha partición. Si la partición no existe deberá mostrar error. Si se elimina la partición

		extendida, deben eliminarse las particiones lógicas que tenga adentro. Recibirá los siguientes valores: **Fast:** Esta opción marca como vacío el espacio en la tabla de particiones. **Full:** Esta opción además marcar como vació el espacio en la tabla de particiones, rellena el espacio con el carácter \0. Si se utiliza otro valor diferente, mostrará un mensaje de error.
-name	Obligatorio	Indicará el nombre de la partición. El nombre no debe repetirse dentro de las particiones de cada disco. Si se va a eliminar, la partición ya debe existir, si no existe debe mostrar un mensaje de error.
-add	Opcional	Este parámetro se utilizará para agregar o quitar espacio de la partición. Puede ser positivo o negativo. Tomará el parámetro **units** para las unidades a agregar o eliminar. En el caso de agregar espacio, deberá comprobar que exista espacio libre después de la partición. En el caso de quitar espacio se debe comprobar que quede espacio en la partición (no espacio negativo).

Ejemplos:

```
#Crea una partición primaria llamada Particion1 de 300kb
#con el peor ajuste en el disco Disco1.disk
```

```
fdisk –Size=300 –path=/home/Disco1.disk  –name=Particion1
#Crea una partición extendida dentro de Disco2 de 300kb
#Tiene el peor ajuste
fdisk -type=E –path=/home/Disco2.disk –U=K -name=Particion2
-size=300
#Crea una partición lógica con el mejor ajuste, llamada Paricion3, #de 1 Mb en el Disco3
fdisk –size=1 -type=L –u=M -f=BF –path="/mis discos/Disco3.disk" –name="Particion3"
#Intenta crear una partición extendida dentro de Disco2 de 200 kb #Debería mostrar error ya que ya existe una partición extendida #dentro de Disco2
fdisk -type=E –path=/home/Disco2.disk –name=Part3 –U=K -size=200
#Elimina de forma rápida una partición llamada Particion1 fdisk -delete=fast -name="Particion1"
-path=/home/Disco1.dsk
#Elimina de forma completa una partición llamada Particion1
fdisk -name=Particion1 –delete=full -path=/home/Disco1.disk
#Quitan 500 Kb de Particion4 en Disco4.dsk
#Ignora los demás parametros (size)
#Se toma como válido el primero que aparezca, en este caso add fdisk -add=-500 –size=10 -u=K
-path="/home/misdiscos/Disco4.disk"
-name=Particion4
#Agrega 1 Mb a la partición Particion4 del  Disco4.dsk
#Se debe validar que haya espacio libre después de la partición fdisk -add=1 -u=M –path="/home/mis discos/Disco4.disk"
-name="Particion 4"
```

4. MOUNT

Este comando montará una partición del disco en el sistema e imprimira en consola un resumen de todas las particiones montadas actualmente.
Cada partición se identificará por un id que tendrá la siguiente estructura utilizando el número de carnet:

*Últimos dos dígitos del Carnet + Numero + Letra

Ejemplo: carnet = 201536958

Id´s = 581A, 581B, 581C, 582A, 583A

Por cada partición montada, deberá imprimir una línea en consola con siguiente formato:
Path del disco utilizado|Nombre de la partición|ID de la partición montada.
El número será el misma para particiones en el mismo disco y la letra diferente para particiones en el mismo disco. (NOTA: Este Comando Debe Realizar el montaje en memoria ram no debe escribir esto en el disco) Los parámetros admitidos por este comando son:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta en la que se encuentra el disco que se montará en el sistema. Este archivo ya debe existir.
-name	Obligatorio	Indica el nombre de la partición a cargar. Si no existe debe mostrar error.

Ejemplo:

```
#Monta las  particione de Disco1.dsk
*canet = 20153695 8
```

mount -path=/home/Disco1.disk -name=Part1
/home/Disco1.disk|Part1|581a

mount -path=/home/Disco2.disk -name=Part1
/home/Disco1.disk|Part1|581a
/home/Disco2.disk|Part1|582a

mount -path=/home/Disco3.disk -name=Part2
/home/Disco1.disk|Part1|581a
/home/Disco2.disk|Part1|582a
/home/DIsco3.disk|Part2|583a

En los ejemplos anteriores, el texto en negrilla es el output del comando mount. Notar que se simula un escenario en el que se ejecutó el comando mount 3 veces seguidas. Por lo que la primera vez, el comando solamente imprime una línea (solo hay una partición montada), la segunda vez imprime 2 líneas, la 3era vez 3 líneas y así sucesivamente. Una línea por cada partición montada exitosamente en el sistema.

5. UMOUNT

Desmonta una partición del sistema. Se utilizará el id que se le asignó a la partición al momento de cargarla. Recibirá los siguientes parámetros:

Parámetro	Categoría	Descripción
-id	Obligatorio	Especifica el id de la partición que se desmontará. Si no existe el id deberá mostrar un error.

Ejemplos:

```
#Desmonta la partición con id vda1 (En Disco1.dsk)
```

```
umount -id=121a
#Si no existe, se debe mostrar error
umount -id=662x
```

6. MKFS

Este comando realiza un formateo completo de la partición, se formateará como ext2 por defecto si en caso el parámetro fs no está definido. También creará un archivo en la raíz llamado users.txt que tendrá los usuarios y contraseñas del sistema de archivos. La estructura de este archivo se explicará más adelante.

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-id	Obligatorio	Indicará el id que se generó con el comando mount. Si no existe mostrará error. Se utilizará para saber la partición y el disco que se utilizará para hacer el sistema de archivos.
-type	Opcional	Indicará que tipo de formateo se realizará. Podrá tener los siguientes valores: **Fast**: en este caso se realizará un formateo rápido. **Full**: en este caso se realizará un formateo completo. Ya que es opcional, se tomará como un formateo

		completo si no se especifica esta opción.
-fs	Opcional	Indica el sistema de archivos a formatear. Podrá tener los siguientes valores: **2fs**: Para el sistema EXT2 **3fs**: Para el sistema EXT3 Por defecto será ext2.

Ejemplos:

```
#Realiza un formateo rápido de la partición en el id 581A en ext2
mkfs -type=fast -id=581A

#Realiza un formateo completo de la partición que ocupa el id 582A
mkfs -id=582A
```

Administración de Usuarios y Grupos:

Este archivo será un archivo de texto, llamado users.txt guardado en el sistema ext2/ext3 de la raíz de cada partición. Existirán dos tipos de registros, unos para grupos y otros para usuarios. Un id 0 significa que el usuario o grupo está **eliminado**, el id de grupo o de usuario irá aumentando según se vayan creando usuarios o grupos. Tendrá la siguiente estructura:

```
GID, Tipo, Grupo
UID, Tipo, Grupo, Usuario, Contraseña
```

El estado ocupará una letra, el tipo otra, el grupo ocupará como máximo **10 letras** al igual que el usuario y la contraseña.

Al inicio existirá un grupo llamado **root**, un usuario **root** y una contraseña (123) para el usuario root. El archivo al inicio debería ser como el siguiente:

```
1, G, root       \n
1, U, root       , root       , 123        \n
```

**Este archivo se podrá modificar con comandos que se explicarán más adelante.*

1. Login

Este comando se utiliza para iniciar sesión en el sistema. No se puede iniciar otra sesión sin haber hecho un **logout** antes, si no, debe mostrar un mensaje de error indicando que debe cerrar sesión. Recibirá los Siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-user	Obligatorio	Especifica el nombre del usuario que iniciará sesión. Si no se encuentra mostrará un mensaje indicando que el usuario no existe. **distinguirá mayúsculas de minúsculas.*
-pwd	Obligatorio	Indicará la contraseña del usuario que iniciara sesión. Si no coincide debe mostrar un mensaje de autenticación fallida. *distinguirá entre mayúsculas y minúsculas.
-id	Obligatorio	Indicará el id de la partición montada de la cual van a iniciar sesión. De lograr iniciar sesión todas las acciones se realizarán sobre este id.

Ejemplos:

```
#Se loguea en el sistema como usuario root
login -usr=root -pwd=123 -id=582A

#Debe dar error porque ya hay un usuario logueado
login -usr="mi usuario" -pwd="mi pwd" -id=582A
```

2. Logout

Este comando se utiliza para cerrar sesión. Debe haber una sesión activa anteriormente para poder utilizarlo, si no, debe mostrar un mensaje de error. **Este comando no recibe parámetros.**

Ejemplos:

```
#Termina la sesión del usuario
Logout

#Si se vuelve a ejecutar deberá mostrar un error
#ya que no hay sesión actualmente
Logout
```

*Todos los siguientes comandos que se explicarán de aquí en adelante, **necesitan que exista una sesión en el sistema ya que se ejecutan sobre la partición en la que inicio sesión**. Si no, debe mostrar un mensaje de error indicando que necesita iniciar sesión.

3. Mkgrp

Este comando creará un grupo para los usuarios de la partición y se guardará en el archivo users.txt de la partición, este comando solo lo puede utilizar el usuario **root**. **Si otro usuario lo intenta ejecutar, deberá mostrar un mensaje de error**, si el grupo a ingresar ya existe deberá mostrar un mensaje de error. Distinguirá entre mayúsculas y minúsculas. Recibirá los Siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-name	Obligatorio	Indicará el nombre que tendrá el grupo

Ejemplo:

```
#Crea el grupo usuarios en la partición de la sesión actual
mkgrp -name=usuarios

mkgrp -name="grupo 1"

#Debe mostrar mensaje de error ya que el grupo ya existe
mkgrp -name="grupo 1"
```

El archivo users.txt debería quedar como el siguiente:

```
1, G, Root    \n
1, U, root    , root    , 123     \n
2, G, usuarios \n
```

4. Rmgrp

Este comando eliminará un grupo para los usuarios de la partición. Solo lo puede utilizar el usuario root, si lo utiliza alguien más debe mostrar un error. Recibirá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-name	Obligatorio	Indicará el nombre del grupo a eliminar. Si el grupo no se encuentra dentro de la partición debe mostrar un error.

Ejemplo:

```
#Elimina el grupo de usuarios en la partición de la sesión actual
rmgrp -name=usuarios

#Debe mostrar mensaje de error ya que el grupo no existe porque ya #fue eliminado
rmgrp -name=usuarios
```

El archivo users.txt debería quedar como el siguiente:

```
1, G, Root     \n
1, U, root     , root     , 123      \n
0, G, usuarios  \n
```

5. Mkusr

Este comando crea un usuario en la partición. Solo lo puede ejecutar el usuario **root**, si lo utiliza otro usuario deberá mostrar un error. Recibirá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-usr	Obligatorio	Indicará el nombre del usuario a crear, si ya existe, deberá mostrar un error indicando que ya existe el usuario. **Máximo: 10 caracteres.**
-pwd	Obligatorio	Indicará la contraseña del usuario. **Máximo: 10 caracteres.**
-grp	Obligatorio	Indicará el grupo al que pertenecerá el usuario. Debe de existir en la partición en la que se está creando el usuario, si no debe mostrar un mensaje de error. **Máximo: 10 caracteres.**

Ejemplo:

```
#Crea usuario user1 en el grupo 'usuarios'
mkusr -usr=user1 -pwd=usuario -grp=usuarios
```

```
#Debe mostrar mensaje de error ya que el usuario ya existe
#independientemente que este en otro grupo
mkusr -usr=user1 -pwd=usuario -grp=usuarios2
```

6. Rmusr

Este comando elimina un usuario en la partición. Solo lo puede ejecutar el usuario **root**, si lo utiliza otro usuario deberá mostrar un error. Recibirá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-usr	Obligatorio	Indicará el nombre del usuario a eliminar. Si el usuario no se encuentra dentro de la partición debe mostrar un error.

Ejemplo:

```
#Elimina el usuario user1
rmusr -usr=user1

#Debe mostrar mensaje de error porque el user1 ya no existe
rmgusr -usr=user1
```

El archivo users.txt debería quedar así:

```
1, G, Root     \n
1, U, root     , root     , 123      \n
2, G, usuarios  \n
0, U, usuarios  , user1     , usuario  \n
```

USUARIO ROOT:

Este usuario es especial y no importando que permisos tenga el archivo o carpeta, el siempre tendrá los **permisos 777** sobre cualquier archivo o carpeta (Esto se explica en detalle

posteriormente). Podrá mover, copiar, eliminar, crear, etc. Todos los archivos o carpetas que desee. No se le negará ninguna operación por permisos, ya que él los tiene todos. Los permisos únicamente se pueden cambiar con *chmod* que se explicará posteriormente.

Se debe tomar en cuenta en que categoría está el usuario, si es el propietario, si pertenece al mismo grupo en que está el propietario o si es otro usuario que no pertenece al grupo del propietario. En base a esta comprobación, el usuario puede estar en tres distintas categorías:

Propietario (**U**)
Grupo (**G**)
Otro (**O**)

Dependiendo de estas categorías se determinan los permisos hacia el archivo o carpeta.

Administración de Carpetas Archivos y Permisos:

Estos comandos permitirán crear archivos y carpetas, así como editarlos, copiarlos, moverlos y eliminarlos. Los permisos serán para el usuario propietario del archivo, para el grupo al que pertenece y para otros usuarios, así como en Linux.

1. Chmod

Este comando cambia los permisos de un archivo o carpeta dentro del sistema de archivos, el usuario que cambia los permisos debe de ser propietario del archivo, los parámetros que recibe son:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta en la que se encuentra el archivo o carpeta a la que se le cambiarán los permisos.

-ugo	Obligatorio	Indica los permisos que tendrán los usuarios. Serán tres números: **U G O** [0-7][0-7][0-7] **U**suario **G**rupo: usuarios dentro del grupo **O**tros: usuarios fuera del grupo. Cada número tendrá los valores desde el 0 al 7. Si el número esta fuera de este rango se mostrará un error. A nivel de bits significan permisos para lectura, escritura y ejecución. Por ejemplo: El número 5 (101) indica permisos para leer y ejecutar. **El número 7 indica permisos para las tres operaciones anteriores.** El número 0 indica que no tendrá permisos para utilizar el archivo.
-r	Opcional	Indica que el cambio será recursivo en el caso de carpetas. El cambio afectará a todos los

		archivos y carpetas en la que la ruta contenga la carpeta especificada por el parámetro **path** y que sean propiedad del usuario actual

Ejemplos:

```
#Cambia los permisos de la carpeta home recursivamente
#Todos los archivos o carpetas que tengan /home cambiarán
#Por ejemplo si existiera /home/user/docs/a.txt
#Cambiaría los permisos de las tres carpetas y del archivo
chmod -path=/home -R -ugo=764


#Cambia los permisos de la carpeta home
#Se debe comprobar que la carpeta home pertenezca al usuario #actual, si no deberá mostrar un mensaje de error.
chmod -path=/home -ugo=777
```

2. touch

Este comando permitirá crear un archivo, **el propietario será el usuario que actualmente ha iniciado sesión**. Tendrá los permisos **664**. El usuario deberá tener el permiso de escritura en la carpeta padre, si no debe mostrar un error. Tendrá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta del archivo que se creará. Si lleva espacios en blanco deberá encerrarse entre comillas. Si ya existe debe mostrar un mensaje si se desea sobreescribir el archivo.

		Si no existen las carpetas padres, debe mostrar error, a menos que se utilice el parámetro **p**, que se explica posteriormente.
-r	Opcional	Si se utiliza este parámetro y las carpetas especificadas por el parámetro **path** no existen, entonces deben crearse las carpetas padres. Si ya existen, no deberá crear las carpetas. No recibirá ningún valor, si lo recibe debe mostrar error.
-size	Opcional	Este parámetro indicará el tamaño en bytes del archivo, el contenido serán números del 0 al 9 cuantas veces sea necesario hasta cumplir el tamaño ingresado. Si no se utiliza este parámetro, el tamaño será 0 bytes. Si es negativo debe mostrar error.
-cont	Opcional	Indicará un archivo en el disco duro de la computadora que tendrá el contenido del archivo. Se utilizará para cargar contenido en el archivo. La ruta ingresada debe existir, si no mostrará un

		mensaje de error.
-stdin	Opcional	Este parámetro es mutuamente exclusivo con el parámetro cont. Parámetro booleano. Indica que el contenido del archivo a crear será obtenido de stdin (standard input).

Si se ingresan los parámetros **cont y **size**, tendrá mayor prioridad el parámetro **cont***

Ejemplos:

```
#Crea el archivo a.txt
#Si no existen las carpetas home user o docs se crean
#El tamaño del archivo es de 15 bytes #El contenido sería:
#012345678901234
touch -SIZE=15 -PatH=/home/user/docs/a.txt –r

#Crea el archivo prometheus.service
touch -stdin -path=/etc/systemd/system/prometheus.service

#Crea "archivo 1.txt" la carpeta "mis documentos" ya debe existir #el tamaño es de 0 bytes
touch -path="/home/mis documentos/archivo 1.txt"



#Crea el archivo b.txt
#El contenido del archivo será el mismo que el archivo b.txt
#que se encuentra en el disco duro de la computadora.

touch -id=vda1 -path=/home/user/docs/b.txt –r -cont=/home/Documents/b.txt
```

3. Cat

Este comando permitirá mostrar el contenido del archivo, si el usuario que actualmente está logueado tiene acceso al **permiso de lectura**. Tendrá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-filen	Obligatorio	Permitirá Admitir como argumentos una lista de n ficheros que hay que enlazar. Estos se encadenarán en el mismo orden en el cual fueron especificados. Si no existe el archivo o no tiene permiso de lectura, debe mostrarse un mensaje de error.

Ejemplos:

```
#Lee el archivo a.txt
Cat –file1=/home/user/docs/a.txt
#En la terminal debería mostrar el contenido, en este ejemplo
#01234567890123

#enlazara los archivos
# a.txt (datos archivo a)
# b.txt (01234567890123)
# c.txt (0123) y debería mostrar el contenido
# siguiente, cada archivo va separado por salto de línea
# datos archivo a
# 01234567890123
# 0123
Cat -file1="/home/a.txt" -file2="/home/b.txt" -file3="/home/c.txt"
```

4. rm

Este comando permitirá eliminar un archivo o carpeta y todo su contenido, si el usuario que actualmente está logueado tiene acceso al permiso de escritura sobre el archivo y en el caso de carpetas, eliminará todos los archivos o subcarpetas en los que el usuario tenga permiso de escritura. Si no pudo eliminar un archivo o subcarpeta dentro de la carpeta por permisos, no deberá eliminar nada dentro de esa carpeta ni la carpeta como tal. Tendrá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta del archivo o carpeta que se eliminará. Si lleva espacios en blanco deberá encerrarse entre comillas. Si no existe el archivo o no tiene permisos de escritura en la carpeta o en el archivo, debe mostrarse un mensaje de error. Si no pudo eliminar algún archivo o carpeta no deberá eliminar los padres.

Ejemplos:

```
#Elimina el archivo a.txt, b.txt muestra error si no tiene permiso
rm -PatH=/home/user/docs/a.txt
rm -PatH=/home/user/docs/b.txt #Error por permisos

#Elimina la carpeta user y todo su contenido (docs, a.txt)
#Si el usuario no tuviera permiso de escritura sobre b.txt
#No debería eliminar las carpetas padres docs ni user, solo a.txt
rm -PatH=/home/user
rm -PatH=/home/user
```

5. Edit

Este comando permitirá editar el contenido de un archivo para asignarle otro contenido**.** Funcionará si el usuario que actualmente está logueado tiene acceso al permiso de lectura y escritura sobre el archivo, si no debe mostrar error. Tendrá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta del archivo o carpeta que se editará. Si lleva espacios en blanco

		deberá encerrarse entre comillas. Si no existe el archivo o no tiene permisos de escritura debe mostrarse un mensaje de error.
-cont	Obligatorio	Contiene la ruta a un archivo en el sistema operativo que contendrá el contenido que será Agregado a la edición.
-stdin	Opcional	Mutuamente excluyente con el parámetro cont. Parámetro booleano. Indica que el contenido a agregar al archivo será leído desde standard input (stdin)

Ejemplos:

```
#Modifica el archivo a.txt
Edit -PatH=/home/user/docs/a.txt -cont=/root/user/files/a.txt
Edit -PatH=/home/user/docs/a.txt -stdin
```

6. Ren

Este comando permitirá cambiar el nombre de un archivo o carpeta, si el usuario actualmente logueado tiene permiso de escritura sobre el archivo o carpeta. Tendrá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta del archivo o carpeta al que se le cambiará el nombre. Si lleva espacios en blanco deberá encerrarse entre comillas. Si no existe el archivo o

		carpeta o no tiene permisos de escritura deberá mostrar un mensaje de error.
-name	Obligatorio	Especificará el nuevo nombre del archivo, debe verificar que no exista un archivo con el mismo nombre, de ser así debe mostrar un mensaje de error.

Ejemplos:

```
#Cambia el nombre del archivo a.txt a b1.txt
ren -PatH=/home/user/docs/a.txt –name=b1.txt

#Deberá mostrar error ya que el archivo b1.txt ya existe
ren -PatH=/home/user/docs/c.txt –name=b1.txt
```

7. Mkdir

Este comando es similar a mkfile, pero no crea archivos, sino carpetas. El propietario será el usuario que actualmente ha iniciado sesión. Tendrá los **permisos 664**. El usuario deberá tener el permiso de escritura en la carpeta padre, si no debe mostrar un error. Tendrá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta de la carpeta que se creará. Si lleva espacios en blanco deberá encerrarse entre comillas. Si no existen las carpetas padres, debe mostrar error, a menos que se utilice el parámetro **p**.

-p	Opcional	Si se utiliza este parámetro y las carpetas padres en el parámetro **path** no existen, entonces deben crearse. Si ya existen, no realizará nada. No recibirá ningún valor, si lo recibe debe mostrar error.

Ejemplos:

```
#Crea la carpeta usac
#Si no existen las carpetas home user o docs se crean
Mkdir –P -path=/home/user/docs/usac

#Crea la carpeta "archivos diciembre"
#La carpeta padre ya debe existir
Mkdir -path="/home/mis documentos/archivos diciembre"
```

8. CP

Este comando permitirá realizar una copia del archivo o carpeta y todo su contenido hacia otro destino.

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta del archivo o carpeta que se desea copiar. Si lleva espacios en blanco deberá encerrarse entre comillas. Debe copiar todos los archivos y carpetas con todo su contenido, a los cuales tenga permiso de lectura. Si no tiene permiso de lectura, no realiza la copia

		únicamente de ese archivo o carpeta. Muestra un error si no existe la ruta.
- dest	Obligatorio	Este parámetro será la ruta a donde se va a copiar el contenido. Debe tener permisos de escritura sobre esta carpeta, si no deberá mostrar un mensaje de error. De no existir la carpeta deberá mostrar un mensaje de error.

Ejemplos:

```
#/
#   home   #664
#          user     #664
#                documents #664
#                       a.txt #664
#                       b.txt   #224
#           images        #664


#Copia documents a images

cp -Path="/home/user/documents" -dest="/home/images"
# b.txt no se copia debido a falta de permisos
#/
#   home   #664
#          user     #664
#                documents #664
#                       a.txt #664
#                       b.txt   #224
#           images        #664
#                documents #664
#                       a.txt #664
```

8. MV

Este comando moverá un archivo o carpeta y todo su contenido hacia otro destino. Si el origen y destino están dentro de la misma partición, solo cambiará las referencias, para que ya no tenga el padre origen sino, el padre destino, y que los padres de la carpeta o archivo ya no tengan como hijo a la carpeta o archivo que se movió. Solo se deberán verificar los permisos de escritura sobre la carpeta o archivo origen.

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta del archivo o carpeta que se desea mover. Si lleva espacios en blanco deberá encerrarse entre comillas. Debe mover todos los archivos y carpetas con todo su contenido, a los cuales tenga permiso de escritura. Si no tiene permiso de escritura, no realiza el movimiento Muestra un error si no existe la ruta.
- dest	Obligatorio	Este parámetro será la ruta de la carpeta a la que se moverá el archivo o carpeta. Debe tener permiso de escritura sobre la carpeta. Si lleva espacios en blanco deberá encerrarse entre comillas. Debe mostrar un mensaje de error si no tiene permisos para escribir o si la carpeta no existe.

Ejemplos:

```
#/
#   home   #664
#          user     #664
#                documents #664
#                       a.txt #664
#                       b.txt   #224
#           images        #664


#Copia documents a images

mv -Path="/home/user/documents" -dest="/home/images"
# mueve b.txt ya que solo se comprueban los permisos
#/
#   home   #664
#          user     #664
#          images #664
#                documents #664
#                       a.txt #664
#                       b.txt   #224
```

9. Find

Este comando permitirá realizar una búsqueda por el nombre del archivo o carpeta. Por cada resultado, imprimirá una linea en consola con el siguiente formato:
nombre archivo|indice de inodo|tipo de archivo.
Donde el tipo de archivo puede ser folder o archivo.
Permitirá los siguientes caracteres especiales:

CARÉCTER	DESCRIPCION
?	Un solo caracter
*	Uno o más caracteres

Recibe los siguiente parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta de la carpeta en el que se inicia la búsqueda, deberá buscar en todo su contenido. Si lleva espacios en blanco deberá encerrarse entre comillas.
-name	Obligatorio	Debe tener permisos de lectura en los archivos que buscará. Indica el nombre de la carpeta o archivo que se desea buscar.

Ejemplos:

find -Path="/" -name=*.js

/home/ren/a.js|256|archivo
/home/ren/ejemplos/b.js|419|archivo
/home/ren/plantillas/node/c.js|169|archivo

10. chown

Cambiará el propietario de uno o varios archivos o carpetas. Lo podrá utilizar el usuario root en **todos** los archivos o carpetas y también lo podrán utilizar otros usuarios, pero **solo** sobre sus **propios** archivos. Recibirá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Este parámetro será la ruta en la que se encuentra el archivo o carpeta a la que se le

		cambiará el propietario. Si no existe la ruta deberá mostrar mensaje de error
- r	opcional	Debe tener permisos de lectura en los archivos que buscará. Indica el nombre de la carpeta o archivo que se desea buscar.
-usr	Obligatorio	Nombre del nuevo propietario, si no existe debe mostrar error

Ejemplos:

```
#Cambia el propietario de la carpeta home recursivamente
chown -path=/home -R -usr=user2

#Cambia los permisos de la carpeta home
Chown -path=/home -usr=user1
```

11. chgrp

Cambiará el grupo al que pertenece el usuario. Únicamente lo podrá utilizar el usuario root. Recibirá los siguientes parámetros:

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-usr	Obligatorio	Especifica el nombre del usuario al que se le cambiará de grupo. Si no existe debe mostrar un error.
-grp	Obligatorio	Contendrá el nombre del nuevo grupo al que pertenecerá el usuario. Si no existe o está eliminado

		debe mostrar un error.

Ejemplos:

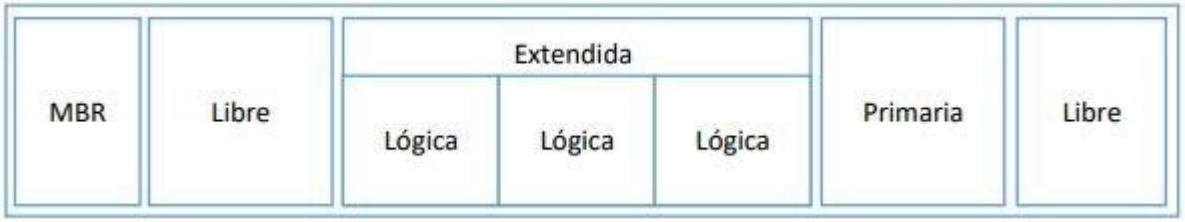
```
#Cambia el grupo del user2
chgrp -usr=user2 -grp=grupo1

#Cambia el grupo del user1
chgrp -usr=user1 -grp=grupo2
```

12. Pause

Este comando será solo la palabra "pause" no tiene atributos al ingresar este comando se pondrá en pausa solicitando que presione cualquier tecla para continuar. Este comando solamente tiene sentido si se encuentra dentro de un archivo script con el comando exec.
Si se utiliza por si solo (sin exec) este comando no realizará ninguna acción.

Discos

Los discos serán archivos binarios que tendrán información del MBR, y espacio con particiones o bien, espacio sin utilizar.
La siguiente figura es un ejemplo de los bloques en un disco con



particiones en el que ya se ha eliminado una partición:

Estructuras

Master Boot Record

Cuando se crea una partición debe utilizarse el primer ajuste para crearla. En la figura anterior debería utilizarse el primer bloque libre para crear una nueva partición. El MBR tendrá los

siguientes campos:

Nombre	**Tipo**	**Descripción**
mbr_tamano	int	Tamaño total del disco en bytes
mbr_fecha_creacion	time	Fecha y hora de creación del disco
mbr_disk_signature	int	Número random, que identifica de forma única a cada disco
disk_fit	char	Tipo de ajuste de la partición. Tendrá los valores **B** (Best), **F** (First) o **W** (worst)
mbr_partition_1	partition	Estructura con información de la partición 1
mbr_partition_2	partition	Estructura con información de la partición 2
mbr_partition_3	partition	Estructura con información de la partición 3
mbr_partition_4	partition	Estructura con información de la partición 4

Partition

Nombre	Tipo	Descripción
part_status	char	Indica si la partición está activa o no
part_type	char	Indica el tipo de partición, primaria o extendida. Tendrá los valores **P** o **E**
part_fit	char	Tipo de ajuste de la partición. Tendrá los valores **B** (Best), **F** (First) o **W** (worst)
part_start	int	Indica en que byte del disco inicia la partición
part_size	int	Contiene el tamaño total de la partición en bytes
part_name	char[16]	Nombre de la partición

Extended Boot Record

Las particiones extendidas tendrán una estructura diferente. Se utilizará una estructura llamada EBR (Extended Boot Record) en forma de lista enlazada, que será como la siguiente:

Nombre	Tipo	Descripción
part_status	char	Indica si la partición está activa o no
part_fit	char	Tipo de ajuste de la partición. Tendrá los valores **B** (Best), **F** (First) o **W** (worst)
part_start	int	Indica en qué byte del disco inicia la partición
part_size	int	Contiene el tamaño total de la partición en bytes.

part_next	int	Byte en el que está el próximo EBR. -1 si no hay siguiente
part_name	char[16]	Nombre de la partición

La estructura lógica de la partición extendida será como la siguiente: El EBR inicial siempre debe existir, aunque se elimine la primera partición. Para crear el archivo del disco se recomienda utilizar un char[1024] como buffer para crear el archivo, si se utiliza un char[1] normalmente se tarda demasiado al momento de crear el archivo.

Pérdida y Recuperación del Sistema de Archivos EXT3

Recovery File System

La recuperación del sistema se hará por medio del journaling y el superbloque. Se recuperará el sistema a un estado consistente antes del último formateo.


```
#Recuperando el sistema de archivos EXT3 de la partición 1
Recovery –id=521A
```

Simulate System Loss

Este formatea los siguientes bloques de datos para simular un fallo en

el disco (una partición en especifica), una inconsistencia o pérdida de información. Se deberán limpiar los siguientes bloques con el carácter /0.

Bloque de bitmap de árbol virtual de directorio
Bloque de bitmap de Bloques
Inodos
Bloque de datos.

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-id	Obligatorio	Especifica el id de la partición a la que se le simulara la pérdida del sistema.

Ejemplo:

```
#Simulando la pérdida del sistema de archivos EXT3 de la
#partición 1
Loss –id=521A
```

Sistema de Archivos EXT2/EXT3

A continuación, se explicarán las estructuras del sistema de archivos EXT2. Se deberán implementar las estructuras como se especifican a continuación.

La estructura en bloques es la siguiente:

El número de bloques será el triple que el número de inodos. El número de inodos y bloques a crear se puede calcular despejando n de la primera ecuación y aplicando la función floor al resultado:

tamaño_particion = sizeof(superblock) + n + 3 * n + n * sizeof(inodos) + 3 * n * sizeof(block)

número_estructuras=floor(n)

El número de bloques será el triple que el número de inodos.
El tamaño del Journaling será de 100 bloques. El número de Journaling, inodos y bloques a crear se puede calcular despejando n de la primera ecuación y aplicando la función floor al resultado:

tamaño_particion = sizeof(superblock) + n + 100*Sizeof(block) + 3 * n + n * sizeof(inodos) + 3 * n * Sizeof(block)

numero_estructuras = floor(n)

Super Bloque

Contiene información sobre la configuración del sistema de archivos. Tendrá los siguientes valores:

NOMBRE	TIPO	DESCRIPCIÓN
s_filesystem_type	int	Guarda el número que identifica el sistema de archivos utilizado
s_inodes_count	int	Guarda el número total de inodos
s_blocks_count	int	Guarda el número total de bloques
s_free_blocks_count	int	Contiene el número de bloques libres
s_free_inodes_count	int	Contiene el número de inodos libres
s_mtime	time	Última fecha en el que el sistema fue montado

s_umtime	time	Última fecha en que el sistema fue desmontado
s_mnt_count	int	Indica cuantas veces se ha montado el sistema
s_magic	int	Valor que identifica al sistema de archivos, tendrá el valor 0xEF53
s_inode_size	int	Tamaño del inodo
s_block_size	int	Tamaño del bloque
s_firts_ino	int	Primer inodo libre
s_first_blo	int	Primer bloque libre
s_bm_inode_start	int	Guardará el inicio del bitmap de inodos
s_bm_block_start	int	Guardará el inicio del bitmap de bloques
s_inode_start	int	Guardará el inicio de la tabla de inodos
s_block_start	int	Guardará el inicio de la tabla de bloques

Esta información estará al inicio del sistema de archivos, no cambia de tamaño y se debe actualizar, según se vayan realizando las operaciones en el sistema de archivos. Por ejemplo, al usuar mount, debe actualizar s_mtime, al utilizar unmount actualizará s_umtime, etc.

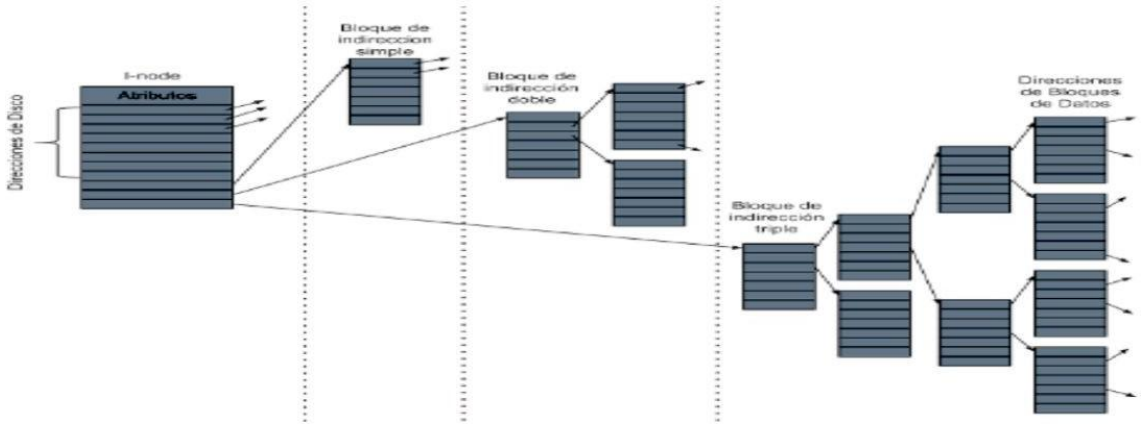
Tabla de inodos

Se utilizará un inodo por carpeta o archivo. Cada inodo tendrá la siguiente información:

NOMBRE	TIPO	DESCRIPCIÓN
i_uid	int	UID del usuario propietario del archivo o carpeta
I_gid	int	GID del grupo al que pertenece el archivo o carpeta.
i_size	int	Tamaño del archivo en bytes
i_atime	time	Última fecha en que se leyó el inodo sin modificarlo
i_ctime	time	Fecha en la que se creó el inodo
i_mtime	time	Úlitma fecha en la que se modificó el inodo
i_block	Int[15]	Array en los que los primeros 12 registros son bloques directos. El 13 será el número del bloque simple indirecto. El 14 será el número del bloque doble indirecto. El 15 será el número del bloque triple indirecto Si no son utilizados tendrá el valor -1.
i_type	char	Indica si es archivo o carpeta. Tendrá los siguientes valores: 1 = Archivo 0 = Carpeta
i_perm	int	Guardará los permisos del archivo o carpeta. Se trabajará a nivel de bits, estará dividido de la siguiente forma: Los primeros tres bits serán para el **U**suario **i_uid**. Los siguientes tres bits serán para el **G**rupo al que pertenece el usuario. Y los últimos

		tres bits serán para los permisos de **O**tros usuarios. Cada grupo de tres bits significa lo siguiente: El primer bit indica el permiso de lectura **R**. El segundo bit indica el permiso de escritura **W**. El tercer bit indica el permiso de ejecución **X**.

La siguiente imagen muestra el funcionamiento de los bloques indirectos.



Bloques de Carpetas

Esta estructura estará asociada a un inodo de carpetas. Aquí se guardará la información sobre el nombre de los archivos que contiene y a que inodo apuntan. La estructura es la siguiente:

NOMBRE	TIPO	DESCRIPCIÓN
b_content	content[4]	Array con el contenido de la carpeta

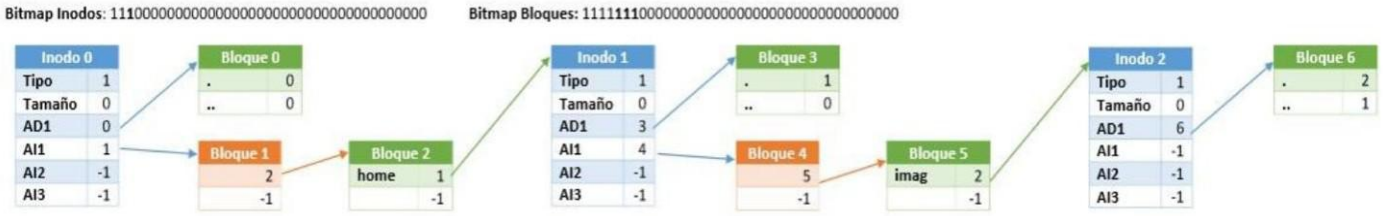
La estructura content será como la siguiente:

NOMBRE	TIPO	DESCRIPCIÓN
b_name	char[12]	Nombre de la carpeta o archivo
b_inodo	int	Apuntador hacia un inodo asociado al archivo o carpeta

El tamaño de este bloque será de 4 * (12 + 4) = 64 bytes. En cada inodo de carpeta, en el primer apuntador directo, en los primeros dos registros se guardará el nombre de la carpeta y su padre.

Ejemplos:

Por motivos de ejemplo se usaron bloques de carpeta (color verde) con capacidad de 2 items, bloques de apuntadores (color anaranjado) con capacidad de 2 apuntadores



Bloques de Archivos

Este bloque guardará el contenido de un archivo. Su estructura es la siguiente:

NOMBRE	TIPO	DESCRIPCIÓN
b_content	char[64]	Array con el contenido del archivo

Por lo que su tamaño, al igual que el bloque de carpetas, es de 64 bytes.

Bloques de Apuntadores

Estos bloques se utilizarán para los apuntadores indirectos (simples, dobles y triples). Su estructura es la siguiente:

NOMBRE	TIPO	DESCRIPCIÓN
b_pointers	int[16]	Array con los apuntadores hacia bloques (de archivo o carpeta)

Su tamaño será de 16 * 4 = 64, igual que los otros dos bloques anteriores.

El tipo de bloque que debe leer o utilizarse se puede determinar según

el tipo de inodo (archivo o carpeta) y en base a que apuntador esté utilizando (directo, simple, doble o triple indirecto)

Limitaciones

En esta sección se calcularán las limitantes del sistema descrito anteriormente. Primero se calculará el máximo de bloques que puede tener asociados un inodo.

numero_bloques_por_inodo = 12 + 16^1 + 16^2 + 16^3 = 12 + 16 + 256 + 4096 = 4380 bloques

Cada bloque de carpeta tiene una capacidad de 4 hijos. Por lo que su capacidad es de 17518 carpetas o archivos dentro de una carpeta.

capacidad_carpeta = 4380 * 4 - 2= 17518

Cada bloque de archivo tiene una capacidad de 64 bytes. Por lo que el tamaño máximo de un archivo es aproximadamente de 273 Kilobytes.

capacidad_archivo = 4380 * 64 = 280320 bytes = 273 Kilobytes

Al formatear se debe crear la carpeta raíz (/) y el archivo users.txt dentro de la raiz.

Otras Operaciones

Aquí se aclararán algunas otras operaciones que se deben realizar. Por ejemplo, que se utilizará el ajuste de la partición para buscar bloques libres y contiguos al momento de crear los archivos.

Al momento de modificar archivos pueden darse tres casos:

Ocupa más espacio: En este caso se utiliza el ajuste de la partición para buscar nuevos bloques contiguos y almacenar el contenido que exceda a los bloques ya utilizados. El contenido se escribe en los bloques que ya se están utilizando y el excedente en los bloques nuevos.

Ocupa menos espacio: Si utiliza menos bloques, únicamente los marcará como libres en el bitmap y eliminará las referencias hacia los bloques en los inodos o bloques de apuntadores

indirectos.

Ocupa igual espacio: Si utiliza la misma cantidad, solo modifica los bloques.

En cualquiera de los casos anteriores debe modificarse el bitmap si es necesario y los datos del inodo (fecha de modificación, etc.)

Si un bloque de apuntadores indirectos queda vacío, se debe marcar como libre en el bitmap y quitar la referencia del inodo o bloque de apuntadores que lo estaba utilizando. Si un archivo ocupa 0 bytes no tendrá bloque asociado.

Cada comando anterior debe modificar las características de los inodos según considere necesario, por ejemplo, un cambio de permisos sobre el archivo modificará el campo i_perm del inodo.

El formateo fast, únicamente limpia con 0s los bitmaps de inodos y bloques. El full aplica un formateo fast y además limpia los bloques e inodos. Siempre debe existir la carpeta raíz y el archivo de usuarios users.txt en la raíz.

SCRIPT

Son archivos con los comandos definidos en este documento. También puede haber comentarios y líneas en blanco. Tendrán la extensión **sh**y se utilizarán para que los ejecute el comando exec.

EXEC

El programa podrá ejecutar scripts con el comando exec. Debe mostrar el contenido de la línea que está leyendo y su resultado. También debe mostrar los comentarios del script.

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-path	Obligatorio	Especifica el nombre del script que se va a ejecutar.

Ejemplo:

```
#ejecuta el script
exec -path=/home/Desktop/calificacion.sh
```

Aclaración

El comando **exec** es **uno de los** comandos que soporta la aplicación, **no el único** comando que soporta. Entiéndase, la aplicación será capaz de ejecutar todos los comandos que se definen en este documento, sin necesitar en ningún momento el comando exec. El comando exec se utiliza como un comando auxiliar, para agilizar la ejecución de varios comandos en sucesión.

REPORTES

Se deberán generar los reportes con el comando **rep**. Se generarán en graphviz. Se puede utilizar html dentro de los reportes si el estudiante lo considera necesario. Deberá mostrarlos de forma similar a los ejemplos mostrados.

IMPORTANTE: Esta parte es obligatoria para tener derecho a la calificación de los aspectos que muestre el reporte. Si falta alguno de los reportes no se calificará.

REP

Recibirá el nombre del reporte que se desea y lo generará con graphviz en una carpeta existente.

PARÁMETRO	CATEGORÍA	DESCRIPCIÓN
-name	Obligatorio	Nombre del reporte a generar. Tendrá los siguientes valores: MBR disk inode Journaling block bm_inode bm_block tree sb file ls Si recibe otro valor que no sea alguno de los anteriores, debe mostrar un error.
-path	Obligatorio	Si recibe otro valor que no sea alguno de los anteriores, debe mostrar un error. Indica una carpeta y el nombre que tendrá el reporte. Si no existe la carpeta, deberá crearla. Si lleva espacios se encerrará entre comillas
-id	Obligatorio	Indica el id de la partición que se utilizará. Si el reporte es sobre la información del disco, se utilizará el disco al que pertenece la partición. Si no existe debe mostrar un error.

-ruta	Opcional	Funcionará para el reporte file y ls. Será el nombre del archivo o carpeta del que se mostrará el reporte. Si no existe muestra error.
-root	Opcional	Para uso exclusivo del reporte Tree. Este parámetro indica el índice del inodo que se utilizará como raíz para el reporte. De no especificarse, se tomará el sistema de archivos entero.

REPORTE MBR

Mostrará tablas con toda la información del MBR, así como de los EBR que se pudieron haber creado.

Ejemplo:

```
rep -id=561A -Path=/home/user/reports/reporte1.jpg -name=mbr
```

MBR Disco1.dsk

Nombre	Valor
mbr_tamaño	10485760
mbr_fecha_creacion	30/11/2015 13:45
mbr_disk_signature	16684811
Disk_fit	W
part_status_1	1
part_type_1	P

part_fit_1	B
part_stax_1	300
part_size_1	1048576
part_name_1	Particion 1
part_status_2	1
part_type_2	E
part_fit_2	W
part_start_2	1048876
part_size_2	1048576
part_name_2	Particion 2

EBR_1

Nombre	Valor
part_status_1	1
part_fit_1	**F**
part_start_1	1048876
part_size_1	524438
part_next_1	1573314
part_name_1	Logica 1

EBR_2

Nombre	Valor
part_status_1	1
part_fit_1	**B**
part_start_1	1573514
part_size_1	524438
part_next_1	-1
part_name_1	Logica 2

REPORTE DISK

Este reporte mostrará la estructura de las particiones, el mbr del disco y el porcentaje que cada partición o espacio libre tiene dentro del disco (La sumatoria de los porcentajes debe de ser 100%).

Ejemplo:

```
rep -id=562A -Path=/home/user/reports/report2.pdf -
name=disk
```

Inode

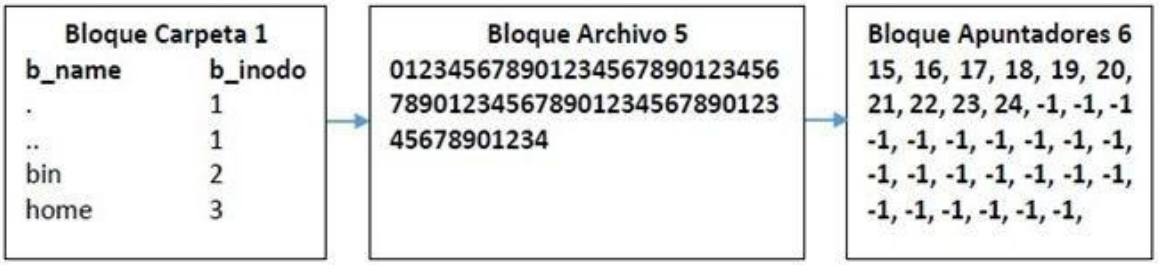
Mostrará información de los inodos **utilizados en modo tabla**. Si no están utilizados no debe mostrarlos.

Index	Tipo	Nombre
0	Folder	/
1	Archivo	/users.txt
2	Archivo	/pwd
3	Folder	/home

Block

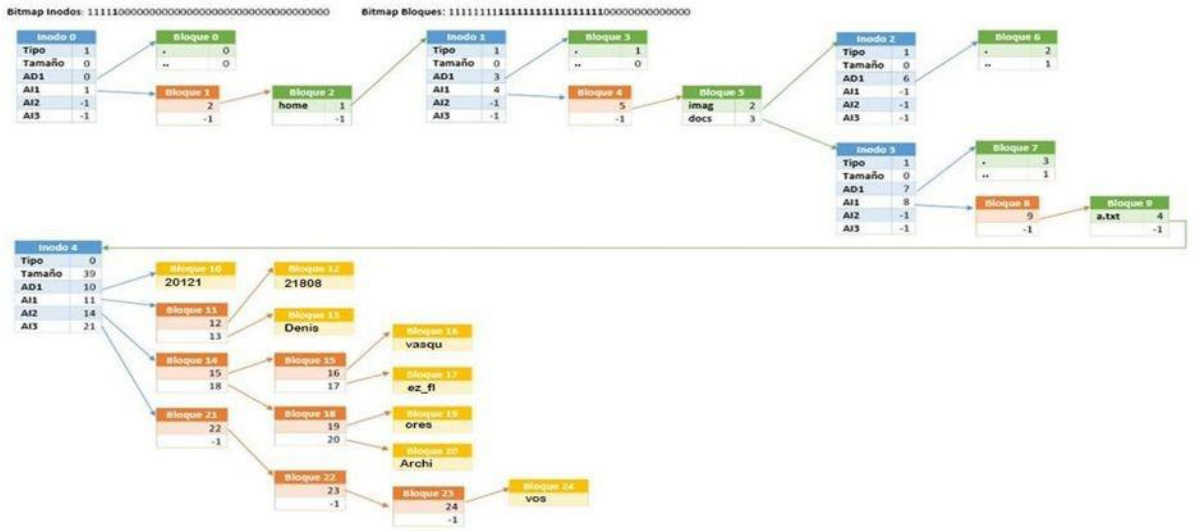
Mostrará la información de todos los bloques **utilizados.** Si no están

utilizados no debe mostrarlos.



Tree

Este reporte genera el árbol de el sistema ext2/ext3. Se mostrará **toda** la información de los inodos o bloques **desde el inodo indicado**. Este comando aceptará el parámetro **root** como un parámetro opcional. De no estar presente, se genera el reporte para todo el sistema de archivos. De indicarse el índice de un inodo, solo deberá generar el reporte del árbol utilizando el inodo inciado como raíz. Si el índice indicado con parámetro root no existe o está vacío, deberá mostar un mensaje de error y no generará el reporte. En este reporte no deben mostarse los bloques o inodos libres, únicamente se mostrarán los bloques que están siendo utilizados. Deberá ser como el siguiente (En este ejemplo no se ponen todos los datos, bloques y flechas por falta de espacio, se utilizaron bloques de carpeta con capacidad 2, bloques de apuntadores con capacidad 2 y bloques de archivo con capacidad 5):



Sb

Muestra toda la información del superbloque en una tabla.

Ejemplo:

SuperBloque Partición 1 en Disco1.dsk

NOMBRE	VALOR
s_inodes_count	200
s_blocks_count	600
s_free_blocks_count	10
s_free_inodes_count	100
s_mtime	17/08/2019 15:38
s_umtime	17/08/2019 15:36
s_mnt_count	4
s_magic	0xEF53
s_inode_size	128
s_block_size	64
s_first_ino	50
s_first_blo	180
s_bm_inode_start	128
s_bm_block_start	328
s_inode_start	630
s_block_start	15852

File

Este reporte muestra el nombre y todo el contenido del archivo especificado en el parámetro file.

Ls

Este reporte mostrará la información de los archivos y carpetas con permisos, propietario, grupo propietario, fecha de modificación, hora de modificación, tipo, fecha de creación.

Permisos	Owner	Grupo	Size (en Bytes)	Fecha	Hora	Tipo	Name
-rw-rw-r--	User1	Mi grupo	40661	24/02/2019	9:53	Archivo	Ejemplo.txt
-rw-r--rwx	User2	Otro grupo	123	20/08/2019	8:13	Carpeta	Home

Documentación

Únicamente se requerirá el manual de usuario. Este deberá ser implementado en lenguaje Markdown y deberán colocarlo en el README de su repositorio. El manual de usuario deberá cubrir la funcionalidad de la aplicación y detallar cada uno de los comandos soportados, su descripción, parámetros y uso.

Entrega

El proyecto se entregará el día **Domingo 05 de septiembre del 2021 hasta las 23:59.** Se utilizará un repositorio de git para que suban su proyecto y se habilitará una opción en UEDI para que puedan subir el link de su repositorio, los auxiliares de cada curso deberán tener acceso a los repositorios respectivos en cualquier momento de la duración del laboratorio, si no se cuenta con acceso se anulara el proyecto, se recomienda que sea un repositorio **privado** para evitar copias. La impuntualidad anulara el trabajo entregado. El servicio de hosting de Git (Github, Gitlab, Bitbucket, etc) queda a discreción del estudiante. Se calificará del último commit que suban a la hora estipulada y se deberá de encontrar dentro del repositorio un ejecutable, desde el cual se calificará. Está prohibido generar el ejecutable durante la calificación.

Para tener derecho a calificación se deberá contar con **requisitos mínimos** los cuales son:

- Aplicación de Comandos directos (sin exec)
- Ejecución de script por medio de exec
- mkdisk
- rmdisk
- fdisk
- mount
- umount
- mkfs
- Reporte MBR y Disk.

El proyecto debe realizarse de forma individual, copias tendrán una nota de 0 y serán reportadas a la escuela.

Las dudas se responderán por UEDI, Classroom, Laboratorios o directamente con el auxiliar por medio de correo electrónico

El lenguaje por utilizar es C/C++. No se permite el uso de otro lenguaje.

Solo se calificará sobre una instalación **física** de una distribución GNU/Linux.

NO se permite la modificación de código durante la calificación. El estudiante únicamente podrá utilizar el ejecutable entregado.

El archivo binario que representa a los discos no debe crecer.

No se permite la utilización de estructuras en memoria (listas, arboles, etc.) para el manejo de los archivos o carpetas.

No se permite agregar o quitar atributos a los structs que se utilizarán en el proyecto.

Para Interpretar la aplicación de comandos es Permitido el Uso de la

Herramienta Bison.